Process to Manage Abbreviations

Virginia Information Technologies Agency
Enterprise Applications Division
Data Management Group

FINAL
Version 1
Revised September 15, 2009

Table of Contents

Introduction

Purpose: The purpose of this document is to define how the VITA Data Management Group will maintain a Standard Abbreviation List. A Standard Abbreviation List is a list of business terms which are abbreviated to produce a consistent, coherent set. These abbreviations are then utilized when creating names for physical database objects.

Scope: This process shall be followed by the VITA Data Management Group and Enterprise Application Development Teams. Executive Branch agencies are welcome to use the VITA Standard Abbreviation List for their development work.

Roles:
The Abbreviation Maker is the person within the VITA Data Management Group who is responsible for maintaining the Standard Abbreviation List. John Morgan is the current Abbreviation Maker (john.morgan@vita.virginia.gov)

Data Modeler in this document refers to anyone who is working with a data model or creating database objects and needs a new abbreviation added to the Standard Abbreviation List.

Why Have Standard Abbreviations?

Although most people reading this document may already be convinced of the value of standard abbreviations for database objects, this brief explanation is included for completeness.

Historically, abbreviations were needed for physical database table and column names because Database Management System (DBMS) products had very restrictive limits on the length of those names. Although most DBMS now allow longer lengths, abbreviations remain desirable because long data names become cumbersome to code and read. Even if the DBMS product in use allows very long data names (such as 128 characters), a best practice is to try to keep names under 30 characters.

Standard abbreviations add value because

1. Modeling tools can apply the abbreviations automatically. This can save time when a logical model is used to generate a physical model and the SQL to create the tables.
2. Modeling tools can do reverse translation and generate meaningful, if not perfect, logical names from the physical names. This helpful functionality only works if the abbreviations are consistent.
3. It improves the clarity of the physical model. If abbreviations are created ad hoc, AR might mean ACCOUNTS RECEIVABLE in one place and ANNUAL REPORT somewhere else. Removing ambiguity makes it possible to look at the column names and have confidence that the meaning is understood.
4. It helps people who write SQL. If all technical and analytical staff who query a database know the standard abbreviations, they can write queries without needing to look up every data name. NUMBER is always NBR, not NUM or NO. FLAG is always FLG, not FL or FLAG.
5. It makes some research possible that would otherwise be difficult or impossible. For example, trying to find all tables which contain Agency Name is easier if all those columns contain “AGCY_NM” plus additional modifiers. This ability facilitates impact analysis.

Management Process and Responsibilities

Abbreviation Maker

- Maintains a list of people who use the Standard Abbreviation List.

Data Modeler

- Creates data model using English words for table and column names.
- Uses a data modeling tool or manually converts names to an abbreviated form.
- Identifies words that are not on the Standard Abbreviation List and therefore could not be abbreviated.
- Sends a request via email to the Abbreviation Maker to create new abbreviations. The email should contain a list of words to be abbreviated and a suggested abbreviation for each word.

Abbreviation Maker

- Reviews requests for new abbreviations.
- Conducts research, as needed.
- Accepts the suggested abbreviation or creates a new one.
- Adds the terms and abbreviations to the Standard Abbreviation List.
- Sends out a communication via email to those using the Standard Abbreviation List.
- Publish new version of Standard Abbreviation List on the VITA website.

Data Modeler(s)

- Update the file, referenced by the data modeling tool, with the latest version of the Standard Abbreviation List.

Creating Abbreviations

Creating abbreviations is inherently a non-deterministic process. That is, two people starting with the same list of words and phrases will come up with different abbreviation lists. Since different agencies have different sets of logical terms, no two agencies would be expected to have identical abbreviation lists. However, each agency shall have a standard abbreviation list and use it on a going forward basis. Agencies without an abbreviation list can start with the VITA Standard Abbreviation List and modify it as appropriate.

The abbreviation lists are expected to be used for in-house database development. There is no expectation that these would be forced on tables and columns which are part of a vendor package.

To enhance readability, examples in this document show physical names in uppercase (PHYS_NM). This is consistent with a DBMS which stores its metadata in uppercase. If the DBMS stores its metadata in lowercase, then the actual names would be in lowercase. If the DBMS supports mixed case (e.g., Sybase/SQL Sever), lowercase with underscores (phys_nm) is preferred over camel case (CmlCse) or uppercase.

Competing goals in creating abbreviations

1. Make physical data names short. The need to abbreviate comes because all DBMS products have size limits on data names. The limit can be a low as 30 for some products but may be as much as 128. Without abbreviations, some data names would even exceed 128 characters.
2. Give each word one and only one abbreviation (mandatory). This contributes to clarity when looking at physical names. For example, it is questionable that CATG_NM and CAT_NM in different tables are the same. They should not be. One might be a “catalog name” and the other a “category name.” If they are both “category name,” the ambiguity requires research to resolve. Unnecessary research is consumes time that could be spent on other tasks. Thus, everyone in the scope of “the enterprise” should use the same abbreviation list.
3. Make the abbreviations understandable to people. Understandability will push toward longer names, which is in tension with goal #1. Moving too far in this direction will cause problems when very long data names appear. In extreme cases, even the abbreviated term can still exceed the limits of the DBMS and force the modeler to create ad hoc physical column names.
4. Make the abbreviations unambiguous. Making all abbreviations unique would eliminate ambiguity but would result in either cryptic abbreviations or longer data names. Thus for terms PEAK, PACK, PACKAGE, PACKAGING, PARK, PARKING, PECK, and PRIMARY KEY, only one could be abbreviated as PK. PCK could be used for PACKAGE and PACKAGING and PACK unless it already

means PICK. Thus some abbreviations will need to be longer to eliminate ambiguity, conflicting with goal #1.

There are two issues related to ambiguity.

a. Can a person look at the physical name and discern the meaning? People become familiar with abbreviation schemes with use so this is not a major issue. People can also discern from the context that PKG_TKT is "parking ticket" and not "package ticket."
b. But, if any process depends upon programmatic reverse translation, confusion will result. If PACKAGE is primary for PKG and PARKING is secondary, then the translation will be PACKAGE TICKET. Depending upon the context this error may be easy or nearly impossible to detect.

5. Allow logical names to be a natural as possible. This goal means that the business should be able to express the logical name as they would normally. If the business uses the term "after-the-fact payment" then you should be able to handle that in your scheme even though the term list may disallow the use of "the" as a term. Consider abbreviating the whole phrase: perhaps AFTER THE FACT → AFTF.

Compromising on abbreviations

Since the goals are in conflict, something has to give. Here are some guidelines:

1. If several words are in a family, they can have the same abbreviation. For example, PACKAGE and PACKAGING could both be abbreviated PKG because they are in the same word family as far as meaning is concerned. This avoids a longer abbreviation like, perhaps, PKGG for PACKAGING. If PKG is used for PACKAGE, then using PKG for PARKING becomes problematic. It is ambiguous for both the human reader and the data modeling tool. Given the data name PKG_FEE is it a packaging fee or a parking fee? What if the system required both a parking fee and a packaging fee to be stored in the same table?

   All modern data modeling tools will convert a logical name to a physical name. And some will convert a physical name back to a logical name. In the latter case, since several terms could have the same abbreviation, the system must choose one as primary and translate all occurrences of an abbreviation to the one primary. Thus for PARKING, PACKAGE and PACKING, if all were PKG, the system would have to make them all, for example, PACKAGE. This is not just ambiguous; it is plain wrong if it should mean PARKING. Another example might be putting VA as the abbreviation for both Virginia and Veterans Administration. When the modeling tool translates from physical to logical, all VAs will become either VIRGINIA or VETERANS ADMINISTRATION. Either way, something is wrong.

   So a clear best practice is: avoid using the same abbreviation for words with no family relationship.

But, what if the legacy abbreviations already violate the best practice? Or what if commonly used acronyms conflict with each other?

Consider the following approach:
Assume the abbreviation list contains both VIRGINIA → VA and VETERANS ADMINISTRATION → VA. Add VA as a term, give it the abbreviation VA and make that primary. Then when the tool does reverse translation, the VA will be left alone and the reader can decide from the context. This allows both terms to use the familiar abbreviation and prevents a wrong reverse translation. The cost is that VA must be interpreted from the context or other metadata.

2. The more frequently a word is used, the more likely it will be end up in a column which has a very long name. Therefore, the more frequently a word is used, the more important it is that its abbreviation be short. Once the name exceeds the limits of the tool, some non-standard compromise is needed and this becomes difficult to manage. Avoid this by leaning toward very short abbreviations for the most commonly used terms.
3. Abbreviate commonly used phrases as a unit. Instead of abbreviating CLOSE OF BUSINESS to CLS_OF_BSNS, put the whole phrase into the abbreviation list and give it its own abbreviation: COB. This will help stay away from hitting the DBMS length limit and still be clear.

   It is important to do this early in the game, ideally before any tables are created using the composite abbreviation. If you add COB to a modeling tool dictionary after columns are named CLS_OF_BSNS, the tool will likely rename the physical names to match the new abbreviation. Now you may have problems with managing the schema because the model does not match the actual database table.
4. If there is a commonly used abbreviation, use that unless other factors dictate otherwise. For example, PRESCRIPTION can be abbreviated RX even though it doesn't follow any of the normal rules.

Steps in creating an abbreviation

1. Is there an abbreviation in common use?
   If APRIL → APR , then what to do with Annual Percentage Rate? ANNPR?
   Or consider the approach described above: use APR for both terms and APR as a separate term and make it the primary term.

2. Is there already a word in the same family? If so, does it make sense to the existing abbreviation? For example if you need to abbreviate CODED and CODE is already abbreviated as CD, you should probably use CD. However, if you need to abbreviate ARBITRATOR and you already have ARBITRATION as ARBTN, it might be confusing to see ARBITRATOR NAME as ARBTN_NM. A new abbreviation, ARBTR, might be in order. Using the existing abbreviation

would also cause a problem if the modeling tool translates the physical name back to ARBITRATION NAME instead of ARBITRATOR NAME.

3. If you need a new abbreviation use the following process:
    a. Right truncate the word. Can the first few characters work as the abbreviation?
    APRIL → APR
    ARBITRATOR → ARB. What about APPLICATION REVIEW BOARD which is already abbreviated ARB? ARB won't work for ARBITRATOR so go for ARBR.
    b. Remove all vowels except a leading vowel:
    GROUP → GRP. That works.
    HEARING → HRNG. That works. What about HIRING? HIR?
    ARBITRATOR → ARBTRTR. Still too long.
    c. Keeping the last letter, remove letters from the right until you are down to four characters: ARBTRTR → ARBR
    Not intuitive enough? Consider adding some characters back.
    d. If the word ends in "ing" try removing the N even if you are already at four characters.
    HEARING → HRNG → HRG
    Make a decision about shortness versus clarity.
    e. Add some letters back. ARBR → ARBTR

4. When the list is nearly complete
    a. Sort the list by logical name and look for changes that make the list as a whole more coherent.
        i. Are there word families sharing an abbreviation which need to be more specific?
        ii. Are there word families which have different abbreviations but could have one abbreviation?
        iii. Are there words in a family which would be good to add preemptively?
    b. Sort the list by abbreviation.
        i. Are there terms with the same abbreviation which would be ambiguous if reverse translated?
        ii. Look for long abbreviations and consider making them shorter. Given a choice, it is better to make high-use terms shorter and low-use terms longer. But, given a choice it is better to make high-use terms intuitive. Going for shortness only AMOUNT would become AM. But, AMT seems more natural and intuitive.

5. Miscellaneous considerations.
    a. If there is a double consonant, remove one of the letters.
    LETTER → LTTR → LTR
    b. For multiple words, does an acronym work?
    OBJECT ORIENTED → OO. Pretty easy.

c. Or create a single abbreviation which is a combination of the two words: AREA HEADQUARTERS → AHQ.
d. If removing a leading vowel leaves the abbreviation phonetically recognizable, then removing the first letter may be acceptable. INVALID → NVLD.
e. If the data modeling tool supports reverse translation, it will have some way to default one of the terms to primary. Its choice may not always be best. Consider:
   PREPARATION → PREP
   PREPARE → PREP
   PREPARER → PREP
   The tool may choose the first one alphabetically as the primary. But, a better choice would be PREPARE because that is the root word. Reviewing and correcting this setting can be time consuming but will reduce confusion over the long haul.
f. If the abbreviation for a longer word (OFFICE) produces a short word (OFF) which can also be used in a model, then the short word should be included in the tools naming file.
   OFFICE → OFF
   OFF → OFF
   The shorter one should be designated as the primary word. Failure to do this will cause automatic reverse translation to yield confusing results.
   OFF SCHECULE FLAG → OFF_SCHED_FLG → OFFICE SCHEDULE FLAG
   OFFICE MANAGER NAME → OFF_MGR_NM → OFFICE MANAGER NAME
   Making OFF the primary would yield the following flow:
   OFF SCHECULE FLAG → OFF_SCHED_FLG → OFF SCHEDULE FLAG
   OFFICE MANAGER NAME → OFF_MGR_NM → OFF MANAGER NAME
   The reader must then determine the meaning of OFF, but at least it is not wrong.
g. If the existing abbreviations already use the same abbreviation for terms in different word families, consider putting the abbreviation in as a term and make it the primary. Say the legacy abbreviations have CREDIT → CR and CIVIL RIGHTS → CR. One solution would say change CIVIL RIGHTS to CIVR and use that going forward. If that is not acceptable, consider adding CR as a term: CR → CR and make it primary. Then, automatic translation will translate something like CR_FRM_NBR to CR FORM NUMBER and the person can try to figure out what CR means. That is better than having it translate to CREDIT when it should be CIVIL RIGHTS or vice versa.

   This can also be a useful technique if an acronym appears that is the same as an existing abbreviation. Put the full term in the list with the acronym as the abbreviation and then add the acronym as a term with the acronym as the abbreviation and make that the primary. Then, both the

original abbreviation and the acronym will remain unchanged in reverse translation. Again, this is better than being plain wrong.

h. Consider the degree to which logical names should be strictly controlled. To allow as much freedom as possible at the logical level, consider adding variants of logical words/phrases and mapping them to the same abbreviation.
   PAY TYPE → PYTP
   PAYMENT TYPE → PYTP
   PAYTYPE → PYTP
   This should be a conscious decision. Choosing not to enter variants in the abbreviation list will allow unintentional variants get into the physical model unabbreviated. You can use this as an aid to enforcing logical names if a disciplined process exists to review physical names looking for words which were not abbreviated.

   To restrict the freedom, some tools allow an administrator to mark some terms as illegal. Be careful. In some environments PAYMENT TYPE and PAY TYPE could have different business meanings. Perhaps PAY TYPE applies to payroll and PAYMENT TYPE applies to bills being paid.

i. Physical data names cannot begin with numerals. Therefore, when abbreviating something like TWELVE HOUR, do not use 12HR. As long as the phrase is in the middle of a term, all is well. But the first time it appears at the beginning, the DBMS will give an error. Example: TWELVE HOUR SHIFT ACCEPTABLE FLAG would become 12HR_SHFT_ACTBL_FLG. Oops! Suggestion: use something like HR12. Everyone will get used to the reversal.